LAHDEN AMMATTIKORKEAKOULU
Lahti University of Applied Sciences

KÄYTTÖLIITTYMÄSUUNNITTELUN NOPEUTTAMINEN

Case: Boström lavausjärjestelmä

LAHDEN
AMMATTIKORKEAKOULU
Tekniikan ala
Tieto- ja tietoliikennetekniikka
Tietoliikenne
Opinnäytetyö
Kevät 2014
Mika Juvonen

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

JUVONEN, MIKA:

Käyttöliittymäsuunnittelun nopeuttaminen
Case: Boström lavauslinja

Tietoliikennetekniikan opinnäytetyö, 39 sivua

Kevät 2014

TIIVISTELMÄ

Tässä opinnäytetyössä on tarkoituksena tutustua käyttöliittymäohjelmistoon ja selvittää kuinka käyttöliittymän suunnittelua olisi mahdollista nopeuttaa. Työssä tutkittiin työn ohella syntyvien komponenttien vaikutusta käyttöliittymän suunnitteluun. Tutkimustyötä varten työssä tutustuttiin iX Developeriin ja sen ominaisuuksiin.

IX Developer on Beijer Electronicsin kehittämä Human Machine Interface -ohjelmisto, jonka avulla saadaan suunniteltua nopeasti käyttöliittymiä. IX Developer mahdollistaa käyttöliittymien kehityksen graafisen suunnittelun lisäksi XAML- ja C# -ohjelmointikielillä.

Työn tavoitteena oli suunnitella iX Developerilla projektipohja, jossa voidaan hyödyntää komponentteja niin, että niitä voitaisiin helposti hyödyntää tulevissa projekteissa, ilman että muutokset tuovat projekteihin merkittävää lisätyötä.

Tutkimustyön tuloksena saatiin aikaiseksi valmiin asiakasprojektin käyttöliittymä, jossa on hyödynnetty modulaarisia komponentteja ja otettu huomioon mahdolliset muutokset tulevissa projekteissa.

Asiasanat: Käyttöliittymäsuunnittelu, Human Machine Interface, Human Computer Interaction, iX Developer, Beijer Electronics

Lahti University of Applied Sciences
Degree Programme in Information Technology

JUVONEN, MIKA:

Rapid development in User Interface designing
Case: Boström assembly line

Bachelor’s Thesis in Telecommunications technology, 39 pages

Autumn 2014

ABSTRACT

This Baclelor's Thesis's purpose is to familiarize in Human Machine Interface software and to determine ways to possibly reduce the time taken between projects in user interface designing. Objectives include creating modular components which accelerate the process of interface designing. To meet the objects, a study of iX Developer and its features was required.

IX Developer is one of the Human Machine Interface softwares developed by Beijer Electronics and it is used for rapid designing of user interfaces in industrial environments. Among its many features, iX Developer also makes it possible to develop your applications with Extensible Application Markup Language and C# -programming languages.

The goal of this thesis was to create a project template with iX Developer which uses modular components. These components would reduce the time taken to create new projects by reducing the required effort in designing them.

As a result an user interface for a customer project was created. This application uses modular components and considers the upcoming changes and requirements future projects might need.

Key words: User interface design, Human Machine Interface, Human Computer Interaction, iX Developer, Beijer Electronics

SISÄLLYS

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena on kehittää Orfer Oy:n käyttöliittymäsuunnittelua tutkimalla, kuinka projekteista saadaan helposti muokattavia ja mukautuvia modulaarisuuden avulla. Tähän mennessä projekteille ei ole ollut valmista pohjaa eivätkä projektit ole olleet helposti muokattavissa uuden projektin tarpeisiin. Opinnäytetyössä käsitellään myös käyttöliittymien suunnitteluun ja kehittämiseen liittyviä periaatteita.

Orfer Oy on vuonna 1970 Orimattilaan perustettu automaatiotekniikan perheyritys. Yrityksen toimialoihin kuuluu robotiikka, tuotejärjestelmien suunnittelu, sekä metallituotteiden valmistus- ja järjestelmien huoltopalvelut. (Orfer Oy 2014.)

Orfer Oy:n käyttöliittymien suunnittelussa käytettävä ohjelmisto iX Developer on Beijer Electronicsin kehittämä Human Machine Interface -sovellus. IX Developer mahdollistaa teollisuudenalan käyttöliittymien nopean ja tehokkaan suunnittelun tarjoamalla käyttäjälle vektoripohjaisten graafisten työkalujen lisäksi myös mahdollisuuden käyttää sovelluksissa ohjelmointia XAML- tai C# -ohjelmointikielillä.

Opinnäytetyön tavoitteena oli kehittää iX Developer -sovelluksella käyttöliittymän suunnittelua nopeuttavia komponentteja, jotka helpottavat uusien projektien luomista valmiin pohjaprojektin avulla. Tavoitteena oli myös saada projekteista helposti muokattavia ja mukautuvia tulevien projektien tarpeisiin.

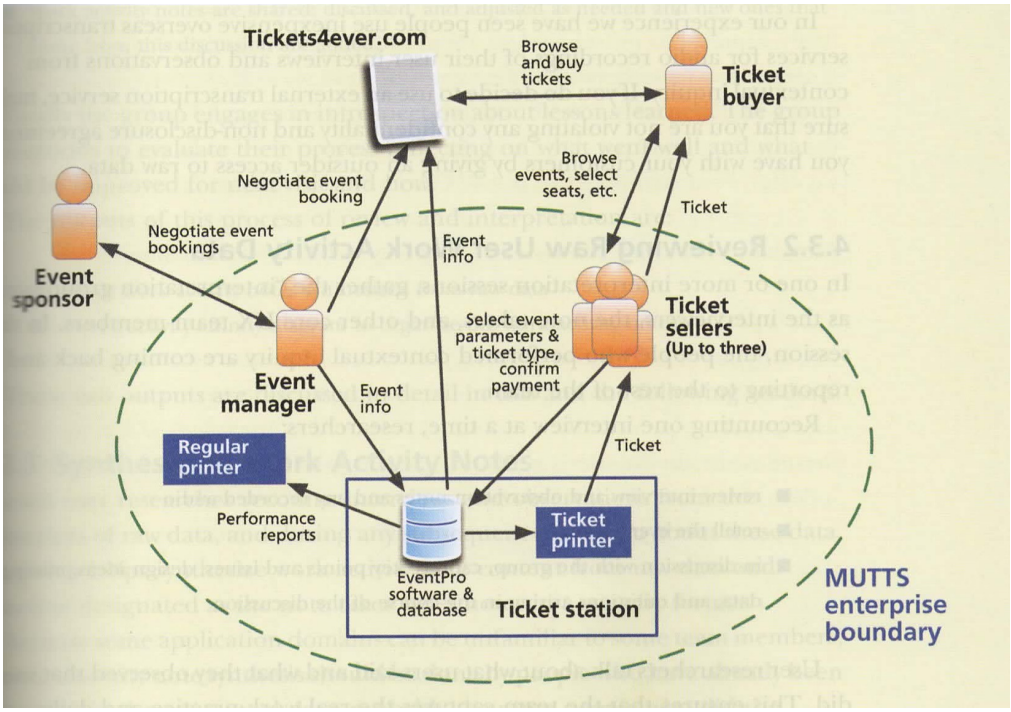
2 KÄYTTÖLIITTYMÄSUUNNITTELU

Käyttöliittymäsuunnittelu voidaan määritellä ihmisen ja tietokoneen välisiä vuorovaikutuksia tutkivan tieteenalan osa-alueiden summana. Tämä kyseinen tieteenala tutkii kuinka ihmiset ja tietokoneet toimivat keskenään ja kuinka käyttäjän tarpeet saadaan tyydytettyä kaikkein tehoikkaimmalla tavalla. (Galitz 2007, 4.) Unger (2012) määrittelee käyttöliittymäsuunnittelun käyttäjäkokemukseen vaikuttavien elementtien luomisena ja synkronoimisena niin, että niillä voidaan vaikuttaa kyseisen käyttäjän havainnointiin ja käyttäytymiseen. Näihin elementteihin voidaan luokitella asiat, joita käyttäjä näkee, kuulee, voi koskettaa, haistaa tai olla muuten vaikutuksissa elementtien kanssa (Galitz 2007, 4; Unger & Chandler 2012, 3).

2.1 Projektin tavoitteiden määrittely

Jokaiselle uudelle projektille määritellään tavoitteet, joiden saavuttaminen ohjaa projektin kulkua. Projektin tavoitteiden tulisi olla samansuuntaisia yrityksen omien tavoitteiden ja toimintasuunnitelman kanssa. Kun tavoitteet ovat selkeitä, on niihin liittyen helpompi esittää olennaisia kysymyksiä, suunnitella käyttäjien tutkimusta ja keskittyä tulosten analysointiin, hioa yksityiskohtia käyttäjien tarpeiden mukaisesti ja täyttää projektin vaatimukset, sekä laittaa projektin vaatimukset tärkeysjärjestykseen. Selkeät tavoitteet auttavat myös valmistautumaan pyyntöihin ja muutoksiin sekä vertaamaan, onko projektille asetetut vaatimukset täytetty tavoitteiden mukaisesti. Selkeä tavoite on helppo ymmärtää, selkeästi erotettavissa muista tavoitteista ja helposti mitattavissa. (Unger & Chandler 2012, 68-69.) Projektin tavoitteita miettiessä on hyvä myös ymmärtää yrityksen vahvuudet ja heikkoudet, tunnistaa mahdollisuudet ja uhat sekä vertailla kilpailijoita. Tätä menettelyä kutsutaan myös SWOT (Strengths, Weaknesses, Opportunities, Threats) -analyysiksi.

Jokaiseen projektiin tarvitaan erilaisten työtehtävien vaativia rooleja, jotka on hyvä terävöittää itselleen heti jokaisen projektin alussa. Yrityksen rooleja voidaan kuvastaa esimerkiksi vuokaavion (Kuvio 1) avulla. Vuokaaviosta ilmenee työntekijöiden roolit ja vuorovaikutukset projektin sisällä, ja siitä selviää yrityksen toiminnan rakenne. Vuokaavion tekeminen helpottaa yleiskuvan hahmottamista projektien välillä ja helpottaa työntekijöiden vuorovaikutusta toistensa kanssa, kun kaikilla on omat, selkeät työtehtävänsä (Unger & Chandler 2012, 30; Hartson & Pyla 2012, 134.)



Kuvio 1: Roolit ja tehtävät yrityksen sisällä (Unger & Chandler 2012, 135)

2.2 Ideointi ja suunnittelu

2.2.1 Käyttäjien kartoittaminen

Projektin ideointi alkaa käyttäjien ja heidän tarpeidensa kartoituksesta eli käyttäjätutkimuksesta. Käyttäjätutkimuksessa yhdistetään haastatteluita ja käyttäjien tarkkailua käyttäjien työympäristössä. Käyttäjiä tarkkailemalla heidän omassa ympäristössään voidaan saada selville esimerkiksi arkipäivän ongelmia, joita käyttäjät kohtaavat, minkälaisella laitteistolla he työskentelevät, mieltymykset hiiren ja näppäimistön tai kosketusnäytön välillä, tai kuinka he työskentelevät muiden työntekijöiden kanssa yhteistyössä (Unger & Chandler 2012, 114-115). Unger (2012) listaa kirjassaan viisi olennaisinta vaihetta käyttäjätutkimuksessa: ensisijaisen käyttäjäryhmän määrittely, suunnitelma käyttäjien lähestymiseen, tutkimuksen suorittaminen, käyttäjäryhmän määritysten varmistaminen tulosten pohjalta ja käyttäjien vaatimusten kartoittaminen.

Käyttäjäryhmät voidaan määritellä tarkasti jokaisen ryhmän kohdalla tai ne voidaan jakaa yksityiskohtaisiin ja visuaalisiin ryhmiin. Ryhmien erottelussa voidaan esimerkiksi listata ominaisuuksia, joilla määritellään erilaiset käyttäjät, käyttää hyödyksi markkinointitilastoja yrityksen asiakkaista tai hyödyntää aikaisemmin suoritettua käyttäjätutkimusta, suorittaa palautekyselyitä tai koostaa asiakaspalveluista usein esiintyviä ongelmia (Unger & Chandler 2012, 102-105). Käyttäjäryhmien erottelun jälkeen asiakkaat ja käyttäjät voidaan eritellä vielä tärkeysjärjestyksen mukaan. Asiakkaiden priorisointi auttaa myös hahmottamaan asiakkaiden vaikutuksen yritykseen paremmin ja kohdistamaan enemmän huomiota tärkeämmille asiakasryhmille.

Käyttäjäryhmien määrittelyn jälkeen on päätettävä, millä tutkimusmenetelmällä käyttäjien kokemuksia aletaan kartoittaa. Lähestymistapoja käyttäjien käyttäytymisen tutkimiseen on monia, mutta Unger (2012, 108-109) on taulukoinut kuusi yleisemmin käytettyä tapaa, sekä milloin niitä on soveliasta käyttää ja mitä vaatimuksia niiden käyttämiseen sisältyy. Näihin kuuteen tutkimusmenetelmään lukeutuvat käyttäjien haastattelut, asiayhteystutkimukset, kyselyt, ryhmäkeskustelut, korttien lajittelu sekä käytettävyystutkimukset.

Haastattelut ovat haastattelijan ja kohderyhmän käyttäjän välisiä keskusteluita, jotka ovat hyödyllisiä kun halutaan saada asiayhteys asiakkaan ja yrityksen välille, mutta asiakkaan toimintaympäristöön ei ole mahdollista päästä. Haastatteluita voidaan tehdä eri kanavien kautta, mutta haastatteluiden ongelmana on saada suoria mielipiteitä ja koota tietoa asenteista tai asiayhteydestä, varsinkin jos haastattelut suoritetaan esimerkiksi puhelimitse. (Unger & Chandler 2012, 111-114.) Jos asiakkaan tiloihin on mahdollista päästä tutustumaan asiakkaan toimintatapoihin ja -ympäristöön, puhutaan asiayhteystutkimuksesta. Tämän tyyppinen tutkimus on hyödyllinen, kun yrityksellä ei ole aikaisempaa tietoa käyttäjien toimintatavoista, tai käyttäjät toimivat erikoistuneessa työympäristössä, kuten esimerkiksi sairaalassa. Ennen asiakaskäyntejä on hyvä ottaa selvää asiakasyrityksestä ja toimintaympäristöstä, ja valmisteltava kysymyksiä käyttäjille sekä ottaa selvää asiakkaan toimintatavoista tai esimerkiksi kilpailusta organisaation sisällä. Asiayhteystutkimuksen haasteena on päästä asiakkaan tiloihin, koska se voi aiheuttaa turvallisuusaiheisia järjestelyitä tai häiritä normaalia työpäivän rytmiä. (Unger & Chandler 2012, 114-117; Hartson & Pyla 2012, 98-104.) Kun halutaan mitattavia tuloksia, kyselyt voivat olla oikea lähestymistapa saada tietoa kohderyhmästä. Kyselyillä saadaan kartoitettua helposti käyttäjien mieltymyksiä, mutta todellisesta suorituskyvystä tai palautteesta ne kertovat hyvin vähän. Kyselyiden haasteena on saada täsmällisiä vastauksia kohderyhmältä johdattelematta vastaajia tiettyyn vaihtoehtoon, joka vääristäisi lopullisia tuloksia. (Unger & Chandler 2012, 118-120.) Kyselyiden sijasta tai ohella voidaan myös käyttää ryhmäkeskusteluita, joissa ryhmän vetäjä ohjaa keskustelua kysymällä osallistujilta tiettyyn aihealueeseen liittyviä kysymyksiä. Ryhmäkeskustelun tavoitteena on selvittää osallistujien tuntemuksia, asenteita ja ideoita keskustellusta aiheesta. Keskustelun haasteena on ymmärtää, kuinka kohdentaa kysymykset niin, että saadaan oikeanlaista tietoa osallistujilta, sekä kuinka ryhmä saadaan toimimaan yhdessä tehokkaasti, jotta keskustelu etenee oikeaan suuntaan. Toisessa ryhmässä suoritettavassa tutkimusmenetelmässä hyödynnetään kortteja, joihin on merkitty esimerkiksi projektiin liittyviä otsikoita, jotka ryhmän jäsenten on luokiteltava joko ennaltamääriteltyihin tai itse keksimiinsä luokkiin. Korttien luokittelulla saadaan jäsenneltyä esimerkiksi projektin sisällön hierarkia, kategoriat ja alikategoriat, joten se on tehokas tapa jäsentää sisältörikkaat projektit helpommin

ymmärrettäviin kategorioihin. Kyseisen tutkimusmenetelmän haasteena on sisällyttää kortteihin parhaat otsikot, jotta rakenteesta saadaan muodostettua järkevä kokonaisuus. (Unger & Chandler 2012, 124-127.) Käyttäjäkokemusta voidaan tutkia myös testaamalla käytettävyyttä, jolloin käyttäjille annetaan oikeudet testata tyypillisiä tehtäviä. Tutkimuksen suorittaja seuraa käyttäjän toimintaa ja voi esittää tarkentavia kysymyksiä käyttäjäkokemukseen liittyen. Käytettävyystutkimusta voidaan hyödyntää, kun olemassaolevaa tuotetta halutaan parantaa ja tuotteelle on olemassa kilpailevia tuotteita. Tutkimuksen haasteena on valita sopivat työtehtävät, jotka käyttäjän tulee suorittaa sekä päättää, kuinka virallinen kyseinen testi tulee olemaan. (Unger & Chandler 2012, 127.)

2.2.2 Sovelluksen vaatimukset

Kun käyttäjät ja heidän toimintaympäristönsä on kartoitettu, alkaa vasta asiakkaan vaatimusten määrittely. Tutkimusmenetelmien avulla saadut tulokset antavat tietoa käyttäjien työympäristöstä ja tavoista, muttei kerro varsinaisesti suunnitteluun liittyviä vaatimuksia. Tulokset on analysoitava ja niistä on koostettava suunnittelun vaatimukset ja malli, ennen kuin itse sovelluksen suunnittelu voi alkaa. Käyttäjien vaatimukset ovat usein muutakin, kuin pelkästään vuorovaikutukseen liittyviä tarpeita, ja niitä voidaan soveltaa laajemmaksi kokonaisuudeksi projektin vaatimuksia suunnitellessa.

Vaatimusten kartoittaminen tuloksista voidaan esimerkiksi tehdä edellisessä kappaleessa mainitulla korttien valintamenetelmällä, isommassa mittakaavassa. Menetelmän ideana on muodostaa hierarkisia ryhmiä esimerkiksi post it-lapuista seinälle. Erivärisillä lapuilla voidaan erotella eri otsikot hierarkian sisällä, ja esimerkiksi värillisellä teipillä merkitä hierarkioiden suhteita toisiinsa. Kyseistä menettelyä kutsutaan myös WAAD (Work Activity Affinity Diagram) -suunnitteluksi. (Hartson & Pyla 2012, 144-157.) Itse prosessissa ryhmä käy läpi WAAD-seinän (Kuvio 2) otsikoita ja ideoita, ja koostaa niistä omasta mielestään sopivia vaatimuksia projektille joko yksin tai pienryhmässä. Vaatimusten on kuitenkin vastattava asiakkaan ja työympäristönsä tarpeita. Ryhmän jäsenten muodostamiin vaatimuksiin kehitetään vastaavat vaatimukset järjestelmältä. Jos asiakas esimerkiksi käyttää monipistekosketusta kosketusnäytöllä, siitä syntyisi

järjestelmälle vaatimukseksi tukea monipistekosketusta sekä miettiä millaisia eleitä asiakas tulee tarvitsemaan järjestelmää käyttäessään.

Kuvio 2: Esimerkki Work Activity Affinity Diagram-suunnittelusta (Jipinghe.com 2012)

2.2.3 Suunnittelun periaatteet

Sovelluksen vaatimusten määrittelyn jälkeen seuraava askel käyttöliittymän suunnittelussa on muodostaa graafinen luonnos käyttäen avuksi suunnittelun perusperiaatteita. Sovelluksen suunnittelussa on otettava huomioon kuinka visuaaliset elementit, vuorovaikutus käyttäjän ja sovelluksen välillä, ja psykologiset tuntemukset vaikuttavat sovelluksen suunnitteluun.

Visuaalisella suunnittelulla on vaikutus käyttäjän ymmärrykseen ja luottamukseen tiettyä merkkiä tai yritystä kohtaan. Visuaalisesti toimiva sovellus vaikuttaa myös käyttäjän alitajuntaan, jolloin käyttäjät aliarvioivat sovelluksen arvon, asiallisuuden tai tärkeyden merkitystä eivätkä tuomitse sovellusta tietoisesti.

Sovelluksen ei tarvitse olla visuaalisesti näyttävä ollakseen toimiva, mutta sen suunnittelussa on hyvä ottaa huomioon muutamia yleisempiä periaatteita.

Sovelluksen elementtien on hyvä olla yhteneviä kaikkialla projektissa sekä tuoda vaihtelua elementtien koon, värin, muodon tai tyylin avulla. Elementtien eroavaisuus tuo sovellukseen vaihtelevuutta, mutta se myös erottelee sovelluksen eri elementit toisistaan ja helpottaa sovelluksen luettavuutta. Sovelluksen ymmärrettävyyttä voidaan myös parantaa vaikuttamalla elementtien hierarkisiin suhteisiin sovelluksessa. Hierarkisella suhteella tarkoitetaan käyttöliittymän suunnittelussa järjestystä, jossa käyttäjä käy elementit läpi. Hierarkialla voidaan ohjata käyttäjän toimia esimerkiksi nostamalla tärkeämpiä elementtejä esiin sovelluksesta, jolloin käyttäjän on helpompi löytää useimmin käytetyt tai tärkeimmät ominaisuudet sovelluksesta ilman, että niiden etsimiseen tai suorittamiseen kuluu liikaa aikaa. Hierarkisiin suhteisiin voidaan esimerkiksi vaikuttaa elementin sijainnilla, värillä, koolla tai tekstin ”äänensävyllä”. (Unger & Chandler 2012, 187-191.) Esimerkiksi painike pelkällä tekstillä ”Paina tästä nähdäksesi tietoja tuotannosta” on vähemmän dominoiva kuin painike, jossa lukee ”Näytä tuotanto” ja johon on lisätty lisäinformaatiosta kertova kuvake.

Toinen tärkeä suunnittelun periaate liittyy käyttäjän ja sovelluksen väliseen vuorovaikutukseen. Vuorovaikutuksella ei pelkästään tarkoiteta, että käyttäjä voi klikata hiirellä tai koskettaa näyttöä, vaan vuorovaikutus lähtee halusta toimia ja ymmärryksestä siitä, mitä käyttäjä voi sovelluksella tehdä. Vuorovaikutus digitaalisessa maailmassa sisältää usein viittauksia fyysiseen maailmaan, joita käyttäjät mieltävät ja yhdistävät toisiinsa. Käyttäjän mieltymykset voivat selkeyttää sovelluksien elementtien toimintaa, mutta voivat myös ohjata käyttäjää väärään suuntaan. Sovelluksessa voitaisiin esimerkiksi käyttää liikenne- tai varoitusmerkeistä tuttuja symboleja ja värejä, mikä kuitenkin edellyttää, että käyttäjä tunnistaa ja erottaa symbolien tarkoituksen oikeasta maailmasta, ja osaa yhdistää symbolien merkityksen sovelluksessa elementin suunniteltuun tarkoitukseen. Yleisenä suunnitteluvirheenä on olettaa käyttäjän viedessä hiiren kursori objektin päälle ja animoimalla kyseinen objekti olevan riittävä osoitus siitä, että objektia voidaan painaa. Tämä toimintatapa olettaa, että käyttäjä liikuttelee hiirtä satunnaisesti toivoen, että jokin objekteista on painettavissa. Hyvin suunnitelluista elementeistä tulee käydä selvästi ilmi, että ne ovat

painettavissa, ilman että käyttäjän tarvitsee kokeilla niitä ensin. (Galitz 2007, 48-50; Unger & Chandler 2012, 196-199.) Käyttäjän ja sovelluksen väliseen vuorovaikutukseen vaikuttaa myös toiminnon suorittamiseen vaadittava aika ja kuinka helposti toiminto voidaan suorittaa. Tähän perustuu myös Fittsin laki, joka arvioi toiminnon suorittamiseen vaadittavaa aikaa kohteen etäisyyden ja koon välisenä funktiona matemaattisen mallin pohjalta. Hyvin suunnitellun sovelluksen elementit ottavat huomioon käyttäjän todennäköisen sijainnin ja suoritettavan toiminnon välisen etäisyyden, siirtymät hiiren tai näppäimistön ja kosketusnäytön välillä, ja kuinka helppoa objektien kanssa on toimia. Sovelluksen helppouteen ja käytettävyyteen vaikuttaa myös toimintoon tarvittavien klikkausten määrä. Tärkeissä ja usein toistuvissa tapahtumissa useamman klikkauksen päässä oleva toiminto on tehotonta ja turhauttavaa käyttäjille. (Unger & Chandler 2012, 200-204.)

Kolmanteen suunnittelun periaatteeseen liittyy käyttäjien psykologinen käyttäytyminen. Psykologisiin tekijöihin luetaan muun muassa visuaalisesti houkuttelevan sovelluksen vaikutus tunteisiin, käyttäjäjien motivaatio sovelluksen käyttämiseen tai sosiaaliset todisteet eli käyttäjien mielipiteet. Houkuttelevan näköisellä sovelluksella voi olla yllättäviäkin tuloksia käyttäjien psykologiseen käyttäytymiseen. Norman (2004, 287-292) on koonnut kirjassaan visuaalisessa suunnittelussa ihmisten käyttäytymiseen vaikuttavia tekijöitä. Havainnoissaan hän huomasi muun muassa, kuinka visuaalisesti houkutteleva sovellus mielletään yleensä helpommaksi käyttää, vaikkei itse sovelluksen käytettävyyteen vaikuteta. Näyttävä sovellus tuo käyttömukavuutta käyttäjän ja sovelluksen väliseen vuorovaikutukseen huolimatta siitä, kuinka nopeasti tai tehokkaasti käyttäjä saa haluamansa toiminnon suoritettua. Visuaalisella sovelluksella voidaan myös rakentaa luottamusta tuotemerkkiin tai yritykseen. Vahvan tuotemerkin avulla voidaan kokeilla käyttäjien tarkkaavaisuutta esimerkiksi tekemällä jotain uudistavaa tai poiketa tutusta kaavasta, ja tarkkailla kuinka muutokset vaikuttavat käyttäjien käyttäytymiseen ja palautteeseen. Luottamuksen rakentamisen ja sovelluksen käyttämisen helpottamisen lisäksi visuaalisuudella on myös merkitys käyttäjän luovuuteen ja positiivisiin tuntemuksiin. Tuotteilla, joilla on rentouttava vaikutus käyttäjiin, on myös vaikutus heidän luovuuteensa. Rentoutuneet ja positiivisesti ajattelevat käyttäjät todennäköisesti tutkivat sovellusta tarkemmin ja

löytävät ratkaisut ongelmiinsa helpommin kuin sovelluksen käytöstä stressaantuneet käyttäjät.

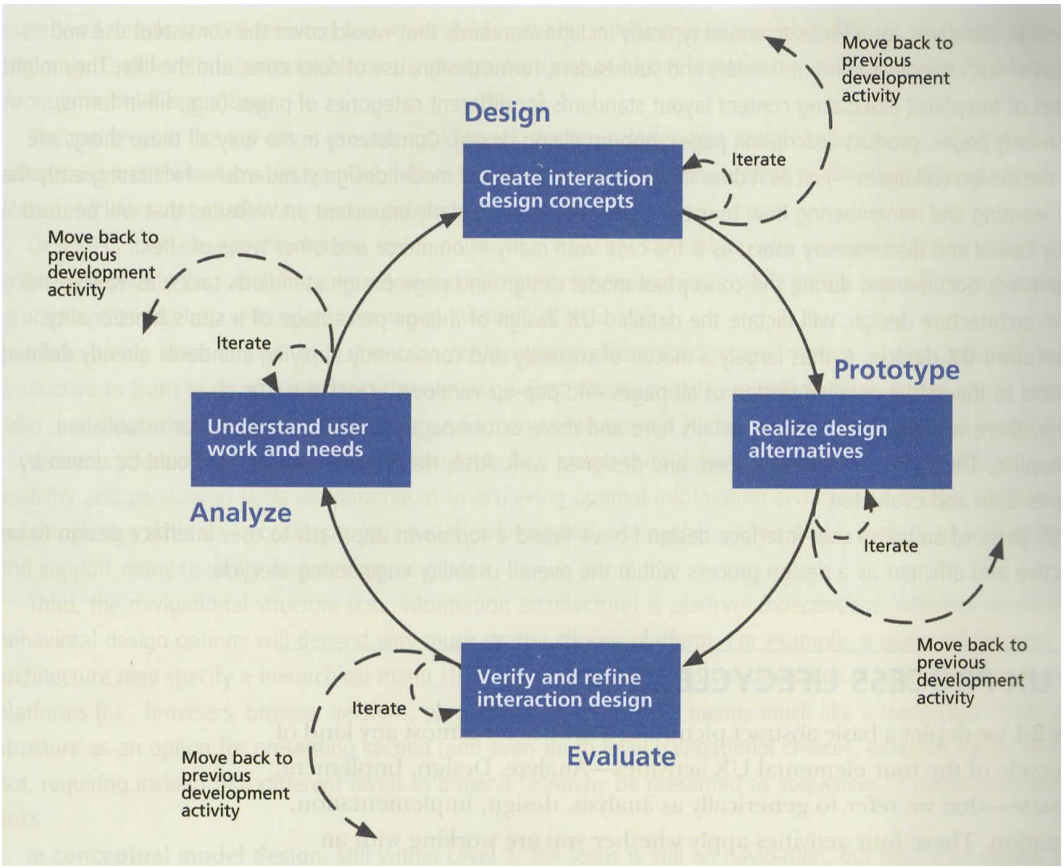
2.3 Toteuttaminen ja kehittäminen

Sovelluksen suunnittelun jälkeen aletaan rakentaa varsinaista sovellusta ja tuottaa sovelluksesta käyttökelpoisia versioita. Tätä jatkuvaa sovelluksen suunnittelun toteuttamista ja kehittämistä kutsutaan myös kehäanalyysiksi. Sovelluksen ensimmäinen versio tai iteraatio syntyy ensimmäisen analyysin ja kehittelyn tuloksena, ja seuraavat versiot pohjautuvat ensimmäisen version iteraatioon, ja näin jatketaan kunnes projekti todetaan valmiiksi käyttöönottoon. Käyttöliittymäsuunnittelu on jatkuvaa kehitystyötä, jonka vaiheet sisältävät sovelluksen analysointia, suunnittelua, käyttöönottoa ja arviointia.

2.3.1 Iteraatiot

Kehitystyöhön on olemassa useita eri lähestymistapoja, joita yhdistelemällä löydetään parhain menetelmä sovelluksen kehitystarpeisiin. Kehityksessä voidaan esimerkiksi käyttää niin sanottua vesiputousmallia, jossa jokaista vaihetta käsitellään erillisenä, ja jossa jokaisen vaiheen vaatimuksena on, että edellinen vaihe on saatu suoritettua ennen kuin aloitetaan uutta. Tämän kehitysmallin ongelmana on kuitenkin oletus siitä, että jokainen vaihe pystytään suorittamaan vähillä muutoksilla. Mikäli tämän mallin vaiheissa tulee muutoksia tai uusia vaatimuksia, on nämä muutokset ensin hyväksyttävä edellisessä vaiheessa, ennen kuin voidaan jatkaa seuraavaan, mikä saattaa ohjata projektin väärään suuntaan alkuperäisestä tavoitteestaan tai aikataulustaan. Toinen, muuntaumiskykyisempi malli ottaa huomioon muutokset vaiheiden välillä. Siinä jokaisen kehäanalyysin jälkeen sovelluksesta tehdään luonnos eli iteraatio. Luonnos syntyy varsinaisen sovelluksen määrittelyn, suunnittelun, kehittämisen ja käyttöönoton tuloksena. Mikäli versio ei kelpaa, se käy uudelleen läpi kaikki kehityksen vaiheet ja näin siitä syntyy sovelluksen toinen iteraatio. Luonnosten tekemistä ja kehittämistä jatketaan niin kauan, kunnes versio on hyväksyttävässä vaiheessa ja valmiina käyttöönottoon. Tämän menetelmän hyvänä puolena on sen mukautuminen muutoksiin ja nopeaan kehitykseen, mutta se vaatii sovelluskehittäjiltä enemmän ryhmätyöskentelyä ja dokumentointia eri versioiden välillä. (Hartson & Pyla 2012, 47-55; Unger & Chandler 2012, 73-82.) Näitä kahta kehitysmenetelmää

yhdistelemällä päästään päättymättömään ympyrämalliin, joka on jatkuvan tuotekehityksen takana oleva perusmalli. Ympyrämallin tarkoituksena on keskittää kehityksen arviointi jokaiseen kehityksen vaiheeseen, joista jokaisesta tehdään iteraatioita tai palataan edelliseen versioon mikäli tarpeellista. Esimerkkipohja tuotekehityksen ympyrämallista on nähtävissä kuviosta 3.



Kuvio 3: Kehityksen ympyrämalli (Hartson & Pyla 2012, 54).

2.3.2 Prototyypit

Käyttökokemuksen ja käyttöliittymien suunnittelussa prototyypit mahdollistavat kokeilla sovelluksen ominaisuuksia ja toiminnallisuuksia käytännössä, ja näin kerätä jatkokehityksen kannalta tärkeää palautetta käyttäjiltä. Prototyypit ovat osa suunnittelun vaiheen iteraavisia prosesseja, joissa tunnistetaan sovelluksen ongelmat ja virheet, tai joissa todennetaan käyttäjän kokemuksia sovelluksesta. Prototyypillä ei vielä tarkoiteta valmiin sovelluksen demottavaa versiota, vaan sen ideana on antaa kuva tulevasta projektista nopeasti ja helposti muokattavassa

muodossa. Prototyypit voidaan luokitella sovelluksen laajuuden tai toiminnallisuuden mukaan, jolloin niistä käytetään käsitteitä horisontaalinen tai vertikaalinen prototyyppi. Horisontaalinen prototyyppi käsittelee valmiin sovelluksen ominaisuuksia laajemmalta alueelta, mutta ei keskity niiden syvällisempään toimintaan, vaan esittelee toimintojen perusrakenteita ja antaa esimakua siitä, minkälaisia ominaisuuksia valmiilta sovellukselta voidaan olettaa. Horisontaalinen lähestymistapa protyyppien kanssa on hyvä tapa aloittaa prototyypit, koska se antaa paremman kokonaiskuvan sovelluksen vaatimuksista ja ominaisuuksista, mutta rajoittaa käyttäjien mahdollisuuksia toiminnallisuuden suhteen. Vertikaaliset prototyypit taas antavat tarkemman kuvan ominaisuuksien toiminnallisuudesta ja keskittyvät enemmän toiminnallisuuteen kuin niiden määrään. Ominaisuuksien pidemmälle viety toiminnallisuus mahdollistaa aidon käyttäjäkokemuksen ja palautteen, mutta vain osasta sovelluksen täysistä ominaisuuksista. Näitä kahta prototyyppikonseptia voidaan yhdistellä niin sanotuksi T-prototyypiksi, jossa kohtaavat täydet ominaisuudet ja tiettyjen ominaisuuksien pitkälle viety toiminnallisuus. Yleisesti projektin prototypointi aloitetaan horisontaalisella prototyypillä, jotta saadaan käsitys sovelluksen kokonaiskuvasta, ja sen jälkeen aletaan rakentaa edistyksellisempää toiminnallisuutta tiettyihin ominaisuuksiin, ja näin sovelluksesta saadaan T-prototyyppi. (Hartson & Pyla 2012, 392-395; Unger & Chandler 2012, 260-261.)

2.3.3 Kehitystyö

Käyttöliittymäsuunnittelun viimeinen vaihe käsittää sovelluksen tai sen prototyypin arviointia ja kehittämistä käyttäjäkokemusten kautta. Sovelluksen jatkokehittämisellä suljetaan suunnittelun periaatteissa käsittävässä kappaleessa mainittu ympyrämalli, mutta samalla sillä avataan uusi tuotekehityksen prosessi. Sovellus ei ole koskaan valmis, vaan sitä kehitetään käyttäjien palautteen ja käyttäkokemukseen pohjautuvilla parannus- tai korjausehdotuksilla.

Käyttäjäkokemus ei ole suoraan mitattavissa eikä sitä voida suoraan vertailla tuloksiin, mutta niistä voidaan tehdä johtopäätöksiä, jotka antavat vihjeitä sovelluksen käytettävyydestä. Esimerkiksi sovelluksen helppokäyttöisyyttä on vaikea mitata suoraan, mutta toimintoihin käytetty aika tai suoritettujen virheiden määrä voi kertoa tarkempia tietoja sovelluksen käytettävyydestä. Samalla tavalla

käyttäjien tyytyväisyyttä ei voida mitata suoraan, mutta sitä voidaan esimerkiksi kartoittaa käyttäjätyytyväisyyskyselyillä. Käyttäjäkokemuksen arvioinnissa käytetään usein harhaanjohdattelevaa termiä käyttäjätestaus, koska käyttäjiä itseään ei arvioida, vaan sovelluksen käytettävyyttä. Käyttäjille tai asiakkaille on tärkeää antaa mielikuva siitä, että he ovat osana käyttöliittymän kehitysprosessia sen sijaan, että heitä itseään arvioitaisiin osana prosessia. (Hartson & Pyla 2012, 427-429.)

Arvioinnissa voidaan käyttää sekä empiirisiä että tieteellisiä tutkimusmenetelmiä tulosten keräämiseksi. Empiirisen arvioinnin tulokset pohjautuvat kokeellisesti määriteltyihin vertailuarvoihin, joihin verrataan käyttäjien kokemuksia sovelluksen käytöstä. Tieteellisen tutkimuksen vertailutulokset taas pohjautuvat sovelluksen mitattaviin vertailuarvoihin ja suorituskykyyn. Empiirinen tutkimus hyödyntää tuloksissaan laadullisia vertailuarvoja, kun taas tieteellinen tutkimus pohjautuu määrällisiin tuloksiin. Laadullinen informaatio määritellään ei-numeeriseksi ja kuvailevaksi informaatioksi, joka antaa tietoa esimerkiksi sovelluksen käytöstä tai sovelluksessa itsessään havaitusta ongelmasta. Määrälliset tulokset taas pohjautuvat numeeriseen dataan, kuten käyttäjän suorituskykyyn tai mielipiteisiin. (Hartson & Pyla 2012, 429-432.)

3 IX DEVELOPER

Beijer Electronics on ruotsalainen teollisuuteen ja automaatiosovelluksiin erikoistuva yritys, joka on tarjonnut teollisuusympäristöihin ja ajoneuvoihin ratkaisuja liittyen tiedonsiirtoon, ohjausjärjestelmiin, moottorikäyttöihin ja visualisointeihin vuodesta 1981 (Beijer Electronics 2014).

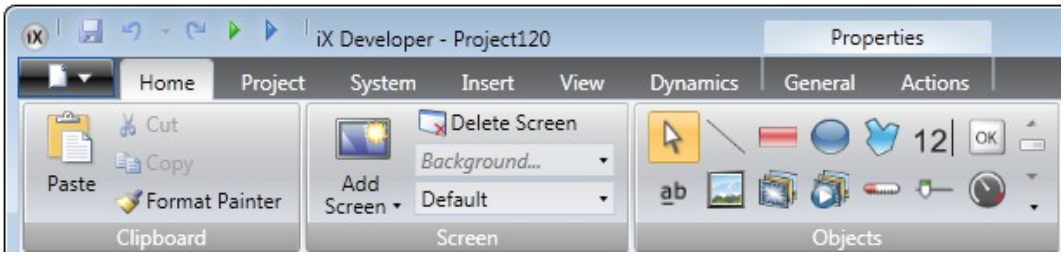
Beijer Electronicsin IX Developer -ohjelmisto on tarkoitettu Human Machine Interface -päätteiden käyttöliittymien nopeaan suunnitteluun, ja sen uusin versio kirjoitushetkellä on 2.10.988. Beijer tarjoaa iX HMI -ratkaisuja kolmessa eri kategoriassa: TxA-, TxB- ja TxC. Jokaisesta sarjasta löytyy kolmea eri näyttökokoa, ja sarjassa ylöspäin mentäessä paneelien ominaisuudet ja käyttökohteet laajenevat. Paneeleissa on esiasennettuna Beijerin iX Runtime, joka vastaa HMI-sovellusten suorituksesta, ja on ohjelmoitavissa iX Developerin avulla.

3.1 Laitteistovaatimukset

iX Developer -kehitysalustaa voidaan käyttää operaattoripaneeleissa, teollisuuskoneissa ja tavallisella PC:llä, jonka käyttöjärjestelmänä on Microsoft XP Service Pack 3, Microsoft Windows Vista tai Microsoft Windows 7. Laitteistolta iX Developer vaatii vähintään 2GB RAM-muistia, 2GHz prosessorin, käyttöjärjestelmänä joko Microsoft Windows XP Service Pack 3, Microsoft Windows 7 tai Microsoft Windows Vista, ja lisäksi näytönohjaimen laitteistokiihdytyksen pitää tukea vähintään pikselinvarjostusta 3. Paneelissa vaatimukset ovat muuten samat, mutta iX Runtime tarvitsee RAM-muistin osalta vähintään 1GB, sekä vähintään 1.3GHz prosessorin. Laitteistovaatimusten lisäksi iX Developer vaatii mediasoittimen käyttöön Windows Media Player versio 10:n, PDF-lukuohjelmaa varten Acrobat Reader versio 9:n ja internetselain tarvitsee toimiakseen Internet Explorer versio 7:n. (Beijer Electronics 2012, 12-13.)

3.2 Päänäkymä

IX Developerin päänäkymä koostuu valikkorivistä (Kuvio 4), projektin selainnäkymästä, työpöytäalueesta ja objektien hallintanäkymästä. Valikkorivi sijaitsee ylimmäisenä, ja sen jokainen välilehti sisältää yhden tai useamman ryhmän, jonka toiminnoilla hallitaan projektin ruutuja ja toimintoja.

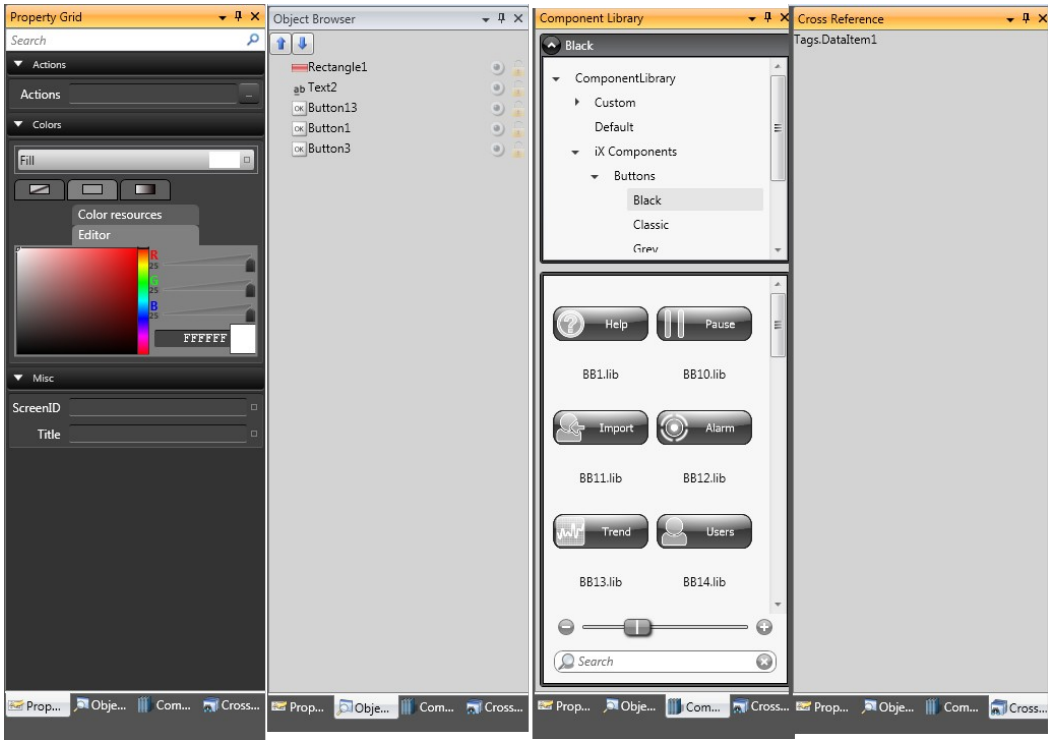


Kuvio 4: iX Developerin valikkorivi

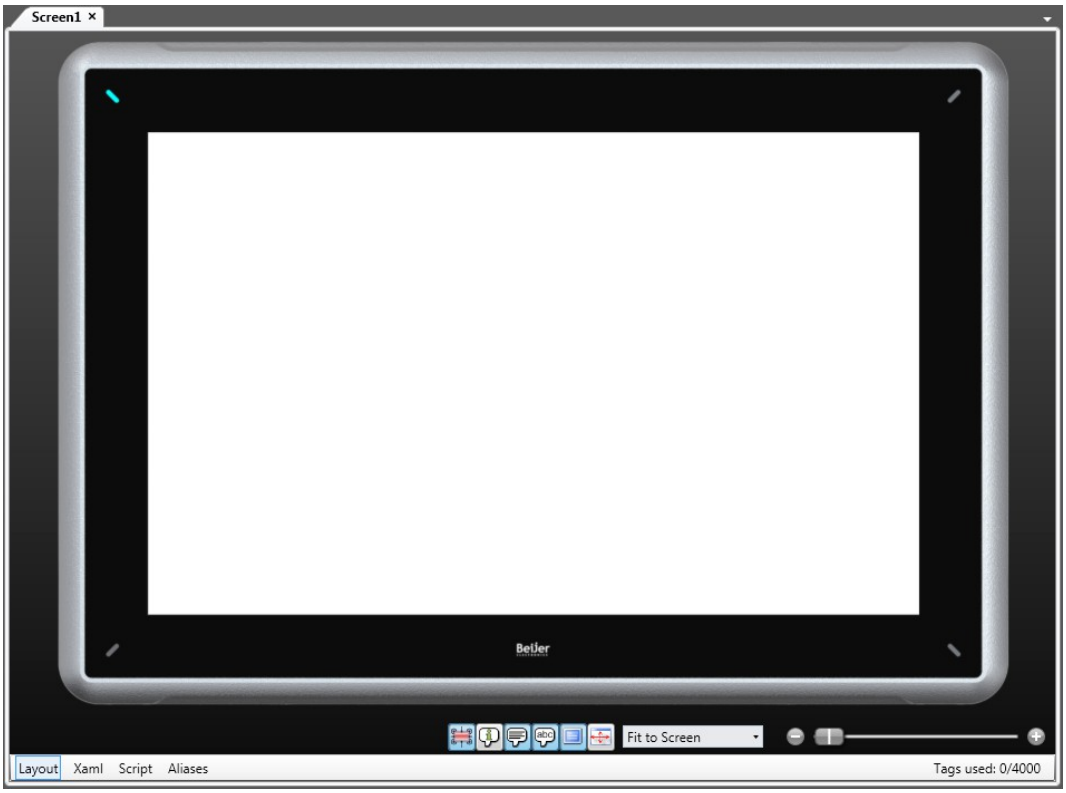
Valikkorivin alla, näkymän vasemmassa laidassa on projektin selainnäkymä (Kuvio 5), jonka avulla voidaan tarkastella projektiin lisättyjä ruutuja, funktioita, reseptikirjastoja, skriptimoduuleita ja tiedonkerääjiä. Jokainen luokka voidaan pienentää näkyvyyden selkeyttämiseksi, ja lisäksi koko selainnäkymä voidaan irrottaa vasemmasta reunasta ja ryhmittää haluttuun paikkaan päänäkymässä.

Päänäkymän oikeassa reunassa sijaitsee ryhmitelty objektien hallintanäkymä (Kuvio 6). Kyseisestä ryhmästä löytyvät toiminnot objektien ominaisuuksien hallinnointiin ja selaukseen, komponenttikirjasto sekä ristiviittaus-välilehti.

Kuvio 5: Projektin selainnäkymä



Kuvio 6: Objektien hallintanäkymä



Kuvio 7: iX:n työpöytänäkymä

Keskelle päänäkymää jää työpöytänäkymä (kuvio 7), jossa näytetään työstettävä ruutu, kyseisen ruudun eri näyttötilat sekä projektin konfigurointinäkymät ja funktiot. Työpöytänäkymän alareunassa on kontrollit, joilla voidaan vaikuttaa ruudun kokoon tai vaikuttaa ruudulla näkyvien vihjeiden ja tietojen näkyvyyteen. Jokaisella ruudulla on myös omat näyttötilansa, joita voidaan vaihtaa näkymän alavasemmasta reunasta. Layout vastaa käyttöliittymän graafista puolta, johon voidaan lisätä sekä iX Developerin valmiita HMI- että Windows Presentation Foundation -objekteja. IX Developer tukee myös objektien ja funktioiden muokkaamista skriptipuolen näkymässä sekä Extensible Application Markup Language että C# -ohjelmointikielillä. Neljäs näkymä on ruutujen alias-näkymä, jossa ruuduille voidaan määrittää alias-ruudut, joiden sisältöä voidaan vaihtaa ilman, että tarvitsee aina luoda uutta kopiota samasta ruudusta.

3.3 Ominaisuudet

Tässä kappaleessa käsitellään osaa iX Developerin tärkeimmistä ominaisuuksista ja toiminnoista sovelluskehityksen kannalta. IX Developerin muista ominaisuuksista löytyy lisätietoa IX:n omasta käsikirjasta (Beijer Electronics 2012).

3.3.1 Objektit

Käyttöliittymien graafinen ulkoasu saadaan aikaan valitsemalla ja luomalla objekteja, joiden käytökseen ja toiminnallisuuteen voidaan vaikuttaa muun muassa toiminnoilla, dynamiikalla ja muuttujilla. IX Developerissa objektit jaetaan eri kategorioihin, joita ovat muodot, virheenkorjausobjektit, HMI-, media-, erikois- ja Windows-kontrollit. Lisäksi IX Developer tukee kolmannen osapuolen objektikirjastoja, joista voidaan tuoda projektiin lisätoiminnallisuutta. Perinteisten objektien lisäksi IX Developer tarjoaa valmiita, vektoripohjaisia kuvia kattavasta komponenttikirjastostaan, johon on myös mahdollista lisätä myös omia objekteja valmiina komponentteina, joihin tallentuvat myös objekteihin lisätyt dynamiikat ja muuttujat. Objektien lisäämisen jälkeen niiden toiminnallisuuteen ja ulkonäköön voidaan vaikuttaa objektien hallintanäkymien tai valikkorivin kontrolliryhmien kautta. Objektien ominaisuusnäkymässä voidaan vaikuttaa esimerkiksi sen väreihin, fyysiseen kokoon, dynamiikkaan ja rajoittaa objektien näkyvyyttä käyttäjille. Objektien selainnäkymässä voidaan tarkastella, järjestellä, uudelleennimetä, rajoittaa näkyvyyttä, lukita objektien muokkaus tai poistaa halutut objektit näkymästä.

Kuvio 8: iX:n valmiit objektit

Lähes kaikille objekteille, muuttujille ja näytöille voidaan määritellä toimintoja, joiden avulla voidaan suorittaa erilaisia funktioita sovelluksessa, ja joiden toimintaa voidaan myös käsitellä skriptien avulla, joista lisää myöhemmissä kappaleissa. Toiminnot määritetään objekteille Actions-välilehden alta, missä valitaan toiminnon aktivointitapahtuma, joka on hiiren tai kosketuspaneelissa sormen liikettä vastaava tapahtuma. Objekteille erilaisia tapahtumia voi olla esimerkiksi Mouse Click, Mouse Up/Down tai Mouse Enter/Leave.

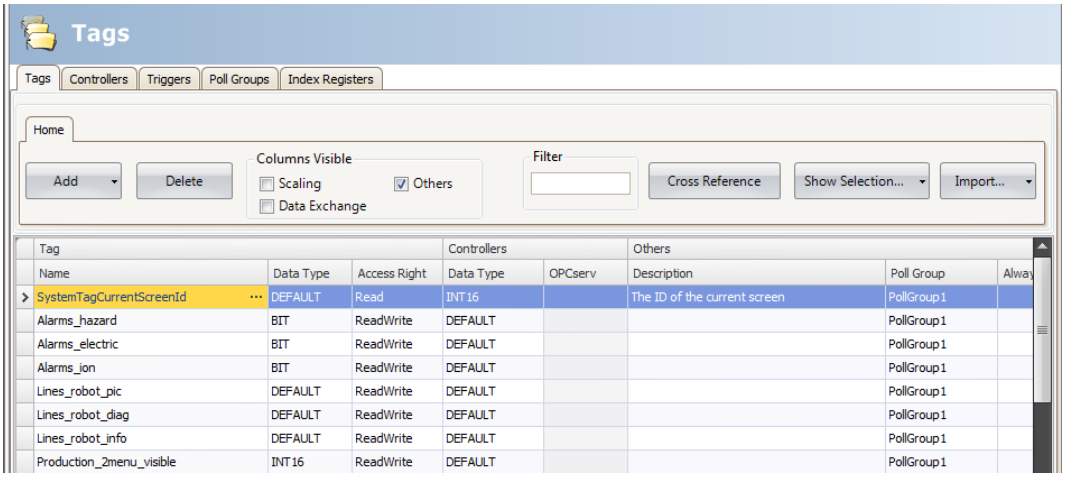
Toimintojen lisäksi objektien ulkonäköä ja muita ominaisuuksia on mahdollista muuttaa dynaamisesti muuttujien avulla. Yhdistämällä objekteihin muuttujia ja vaihtamalla näiden arvoja, voidaan vaikuttaa esimerkiksi objektin kokoon, sijaintiin ja näyttö- tai tekstikenttien arvojen muuttamiseen.

Kaikille objekteille yhteistä on valikkorivin General-välilehti. General-välilehdellä (kuvio 9) voidaan määritellä objektin kuvien ja tekstin kohdistus, tyyli sekä vaikuttaa näkyvyyteen. Mahdollisten ominaisuuksien määrä riippuu valitusta objektista ja sen tyypistä.

Kuvio 9: Valikkorivin General-välilehti

3.3.2 Muuttujat

IX Developerissa on kolmenlaisia muuttujia: sisäiset, ulkoiset ja järjestelmämuuttujat. Sisäiset muuttujat ovat sovelluksen paikallisia muuttujia, ja niitä voidaan käyttää esimerkiksi arvojen näyttämiseen, kirjoittamiseen ja lukuun, tai välimuuttujina, joihin tallennetaan ulkoisten muuttujien arvoja, joita käytetään taas johonkin muuhun funktioon projektissa. Ulkoiset muuttujat tulevat ohjelmalogiikalta, ja ne vastaavat tiettyjä muistipaikkoja logiikassa. Kolmas muuttujaryhmä, järjestelmämuuttujat, ovat järjestelmän tietoja näyttäviä muuttujia. Järjestelmämuuttujien avulla voidaan esimerkiksi näyttää vapaana olevan muistin määrä, viimeksi kirjautuneen käyttäjän nimi, tai tietoliikennevirheet. Muuttujien määritykset tehdään iX Developerin funktioissa, kohdassa Tags. Tags-funktion avulla muuttujille voidaan määritellä tietotyyppi ja lukuoikeudet, kytkeä muuttujat logiikkaohjaimiin, vaihtaa muuttujien kyselyryhmää ja määrittää muuttujille toimintoja, jotka aktivoituvat esimerkiksi muuttujan arvon muuttuessa.



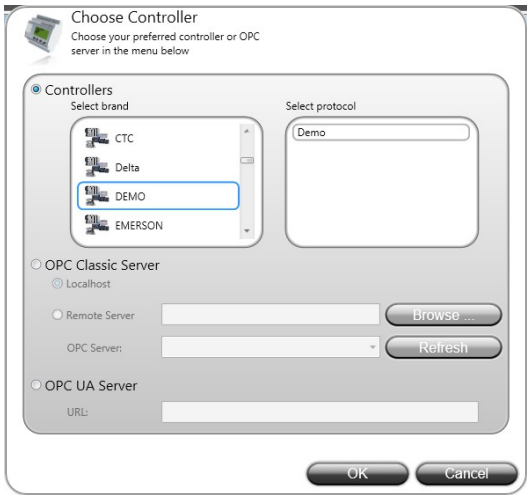
Kuvio 10: Muuttujien määritys

Muuttujat ovat tärkeä osa iX Developerin sovelluskehitystä, koska niillä ohjataan sovelluksessa ja logiikassa käytettäviä arvoja. Muuttujia voidaan esimerkiksi käyttää objekteissa, skripteissä, hälytyksien luomisessa, sisäisissä muuttujien muunnoksissa tai tekstikirjastoissa. Niillä voidaan vaikuttaa objektien näkyvyyteen ja ulkonäköön muuttamalla objektien dynamiikkaa, tai niitä voidaan käyttää vertailuehdoissa, joiden toteutuessa suoritetaan haluttu toiminto.

Muuttujia voidaan tuoda iX Developeriin .csv -tekstitiedostoina helppoa lisäystä varten, ja ne voidaan myös tarvittaessa ottaa talteen tekstitiedostoon, joita voidaan myös muokata Microsoftin Excel -laskentataulukossa tai tekstieditorissa. Tekstitiedostojen lisäksi muuttujia voidaan tuoda logiikkaohjaimilta tai niihin voidaan viedä muuttujia muuttujalistojen avulla. Muuttujia tuodessa iX antaa käyttäjälle valita, mitkä muuttujat kyseisestä tiedostosta tuodaan. Muuttujien nimet ja muut kentät on tuomisvaiheessa sidottava oikein, jotta iX osaa yhdistää muuttujalistan arvot omiinsa.

3.3.3 Logiikkaohjaimet

Sisäisten muuttujien lisäksi iX Developer hyödyntää logiikkaohjainten ulkoisia muuttujia. Logiikkaohjaimia voidaan lisätä ja muuttaa niiden asetuksia Tags-funktion Controllers-osiossa. Ohjainta lisätessä on mahdollista valita kolmesta vaihtoehdosta: fyysinen logiikkaohjain, OPC Classic Server tai OPC UA Server (kuvio 11).

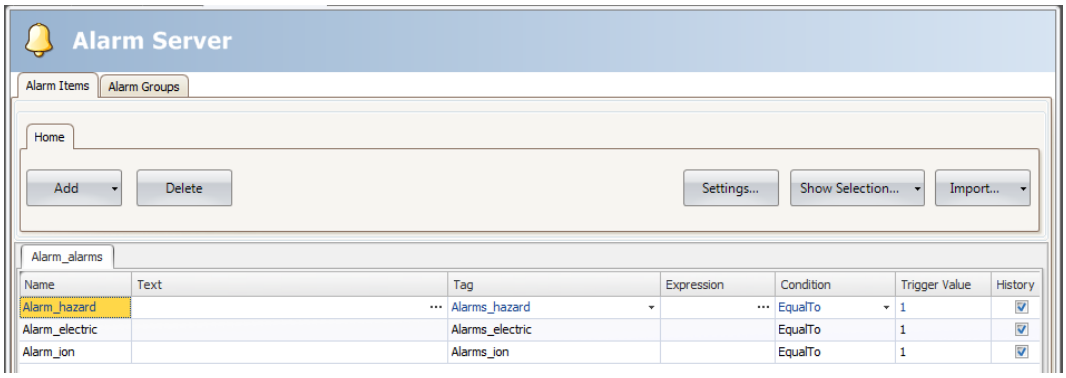


Kuvio 11: Logiikkaohjaimen lisääminen

Lyhenne OPC tulee sanoista OLE (Object Linking and Embedding) for Process Control ja sillä tarkoitetaan ohjelmistorajapintaa, jonka avulla Windows-käyttöjärjestelmä pystyy kommunikoimaan teollisuuslaitteiden kanssa. OPC koostuu palvelin/asiakas-pareista, jossa palvelin vastaa Programmable Logic Controllerilta eli logiikkaohjailmelta tulevien tietojen muuntamisesta OPC-protokollan mukaisiksi tiedoiksi. Nämä tiedot välitetään OPC asiakasohjelman välityksellä HMI-sovellukselle, tässä tapauksessa iX Developerille. (Cogent Real-Time Systems 2010; Beijer Electronics 2012, 140.) OPC Unified Architecture on uuden sukupolven teknologia, jonka tarkoituksena on korvata vanhentunut OPC-protokolla. OPC UA on selainpalveluita hyödyntävä, laitteistoriippumaton palvelin-asiakaslähtöinen standardi, jonka avulla laitteet voivat kommunikoida erilaisten verkkojen välillä. Se tarjoaa parempaa tietoturvaa ja luotettavampaa yhteyttä verrattuna edeltäjäänsä, joka perustui vanhentuneeseen Microsoft-COM/DCOM-teknologiaan. (IEC 62541 2010; Mahnke & Leitner 2009, 57.)

3.3.4 Hälytykset

Hälytykset määritellään iX Developerissa Alarm Server-funktion avulla. Hälytykset kytketään muuttujiin, joille määritellään vertailuehdot hälytyksen toteutumiselle. Hälytyksiä tarkastellaan Alarm Viewer -objektin avulla, josta näkyy hälytyksen tila, nimi ja kuvaus. Objektista löytyy toiminnot hälytysten kuittaamiseen, poistamiseen tai suodattamiseen, ja lisäksi sen avulla on mahdollista tarkastella hälytyksen tarkempia tietoja.



Kuvio 12: Hälytyksien määrittäminen

Hälytyksillä on iX Developerissa neljä mahdollista tilaa: Active, Inactive, Acknowledged ja Normal. Kun hälytys on aktiivinen ja sitä ei ole kuitattu, näkyy se Alarm Viewerissä aktiivisena, ja näytölle ilmestyy hälytystä kuvaava symboli, joka voidaan myös poistaa käytöstä iX Developerin asetuksista. Hälytyksen tila muuttuu aktiivisesta passiiviseksi, kun hälytystä vastaavan muuttujan arvo on palautunut normaaliin tilaansa, mutta hälytystä ei ole vielä kuitattu. Kun hälytys kuitataan sen ollessa vielä aktiivinen, sen tilaksi tulee Acknowledged. Normaalitilassa hälytys on kuitattu ja se on palautunut normaaliin tilaansa.

Hälytykset on mahdollista lähettää eteenpäin joko tekstiviestinä, sähköpostilla tai tulostaa paikallisella tulostimella hälytysten jakelukeskuksen kautta. Funktiota voidaan käyttää joko projektin sisäisesti tai määrittää yksi hälytyspalvelin, joka toimii hälytysten välittäjänä asiakaskoneille hälytysverkossa. Hälytykset tallennetaan myös paikalliseen tietokantaan, jotta hälytykset eivät häviä tietokannasta esimerkiksi sähkökatkoksen aikana.

3.3.5 Tietokannat

Projektin käännön aikana iX Developer päivittää paikallisen tietokantansa Database.sdf -tiedostoon, joka on SQL Server Compact Edition -tietokanta. Tietokannan avaamista ja tarkastelua varten voidaan käyttää iX Developerin asennuksen mukana tulevaa Database Vieweriä, vastaavaa objektia, tai kolmannen osapuolen sovelluksia, jotka tukevat SQL Server Compact Edition -tietokantoja. Tietokanta on koottu useista projektin eri funktioiden käyttämistä tietokannoista, joihin lukeutuvat resepti-, tiedonkeruu-, käyttäjäloki- ja hälytystietokannat. Näistä ainoastaan reseptitietokantaa voidaan suoraan muokata sovelluksessa, kaikki muut tietokannat tulevat automaattisesti sovelluksesta käyttäjän lisäämien funktioiden mukaisesti.

Reseptit luodaan Insert-välilehden alta lisäämällä Recipes -funktio, jossa reseptien lisääminen, poistaminen ja muokkaaminen tapahtuu. Jokaiselle reseptin osalle on määriteltävä nimi ja muuttuja, jota reseptin osa vastaa. Tämän jälkeen voidaan luoda valmiita muuttujien arvojen variaatioita, eli reseptejä, joihin määritellään kunkin reseptin muuttujan arvo.

Data Logger on funktio, jota käytetään iX Developerissa haluttujen muuttujien arvojen keräämiseen. Arvoja voidaan kerätä tietyn ajan välein tai muuttujan arvon muuttuessa, ja niitä voidaan näyttää Trend Viewer -kuvaajina, tai tarkastella sellaisenaan tietokannassa.

Audit Trail -tietokantaan tallentuu käyttäjän tekemät muutokset ja toiminnot projektin ajon aikana, joten se toimii iX Developerissa käyttäjälokina. Funktiota voidaan käyttää esimerkiksi vianselvitystilanteissa käyttäjän toiminnan jälkeen. Tietokantaa voidaan tarkastella Audit Trail Viewer -objektin avulla, tai tietokanta voidaan tuoda .sdf-tiedostona myöhempää tarkastelua varten.

3.3.6 Raportit

IX Developerissa on mahdollisuus luoda raportteja, joihin voidaan esimerkiksi kerätä tilastollisia tietoja tuotannosta, käyttäjän toimintoja tai luoda kuvaajia. Raportteja varten on luotava ensin iX Developerin syntaksin mukainen raporttipohja, joka voidaan lähettää tulostimelle tai se voidaan tallentaa koneelle .xls -tiedostomuodossa. Raportissa näytettävien muuttujien arvot määritellään raporttipohjiin korvaavilla kentillä, joihin tiedot siirtyvät projektin ajon eli iX Runtimen aikana. Muuttujat määritellään .xls tiedostoon kentällä <#Tag(muuttujan nimi)>. Kuviossa X on esimerkki valmiista raportin mallipohjasta. Valmis raportti pohja tulee myös sisällyttää projektin tiedostoihin, jotta sitä voidaan käyttää. (Beijer Electronics 2012, 324.)

	A	B	C	D	E
1	**Basic Report. Show momentary values**				
2	**<#TAG(SystemTagDateTime)>**				
3					
4	Value1	<#TAG(Value1)>			
5	Value2	<#TAG(Value2)>			
6	Value3	<#TAG(Value3)>			
7	Value4	<#TAG(Value4)>			
8	Value5	<#TAG(Value5)>			
9					
10	**Sum**	**#ARVO!**			

Kuvio 13: Esimerkki valmiista rapottipohjasta

Raporttien luominen ei rajoitu pelkästään muuttujia sisältäviin tiedostoihin. Raportissa näytettäviä tietoja voidaan myös hakea projektin tietokannoista käyttämällä raporttipohjissa SQL-tietokantakyselyitä. Tietokantapohjissa käytetään samaa raporttipohjaa kuten normaalisti, mutta siihen on lisättävä uusi <#Config> -niminen välilehti. SQL-kyselyt alkavat riviltä 10, ja käsittävät kaksi saraketta A ja B, joissa kyselyt suoritetaan. Sarakkeella A määritetään kyselyn nimi ja vastaavasti sarakkeessa B määritellään itse hakukomento. Sarakkeen A nimiä käytetään muualla tiedostossa vastaamaan haettuja tuloksia, ja niihin viitataan vastaavalla syntaksilla kuin tavallisissa raporttipohjassakin, lisäyksenä tietokannan sarakkeen nimi: <#Tietokantahaun nimi.Tietokannan sarake>. (Beijer

Electronics 2012, 325-326.) Malli valmiista tietokantaraportista ja sivujen määrityksistä on nähtävissä kuvioista 14 ja 15.

	A	B
1	**Configuration Sheet**	
2		
3	NOTE!	
4	Configuration information can not start above row 10	
5		
6	DataSet (*.xsd)	
7		
8	Data	
9	SQL QueryName Used in the other sheets	SQL Query to iX
10	DataLoggerCurrentMonth	SQL(General; Select CONVERT(datetime, CONVERT(nvarchar(10), Time, 103),103) as Date, sum(LogItem1) as LogItem1, sum(LogItem2) as LogItem2, sum(LogItem1) as LogItem3, sum(LogItem1) as LogItem4, sum(LogItem1) as LogItem5 FROM Datalogger1 WHERE Time <= GetDate() AND DatePart(mm,Time) = DatePart(mm,GetDate()) GROUP BY CONVERT(datetime, CONVERT(nvarchar(10), Time, 103),103))
11	DataLoggerTop25	SQL(General; select top(25) * from DataLogger1 order by time desc)
12	AuditTrailTop25	SQL(AuditTrail; select top(25) * from AuditLog order by TimeStamp desc)

Kuvio 14: Tietokantaraportin Config-tiedoston määrittely

	A	B	C
1	**Latest 25 Items**		
2			
3	**Date**	**LogItem1**	**LogItem2**
4	<#DataLoggerTop25.Time>	<#DataLoggerTop25.LogItem1>	<#DataLoggerTop25.LogItem2>
5			
6	**Sum:**	0,00	0,00

Kuvio 15: Esimerkki tietokantaraportin mallista

Raporttipohjien määrityksen jälkeen raportin luominen tapahtuu objektin toimintoon lisätyn Generate Report-toiminnon avulla. Raportin luominen ei välttämättä tapahdu heti, ja riippuu käsiteltävien muuttujien määrästä. Lisäksi raporttipohjia, joissa on kaavioita tai kuvaajia, ei voida suoraan tulostaa operaattoripaneeleista.

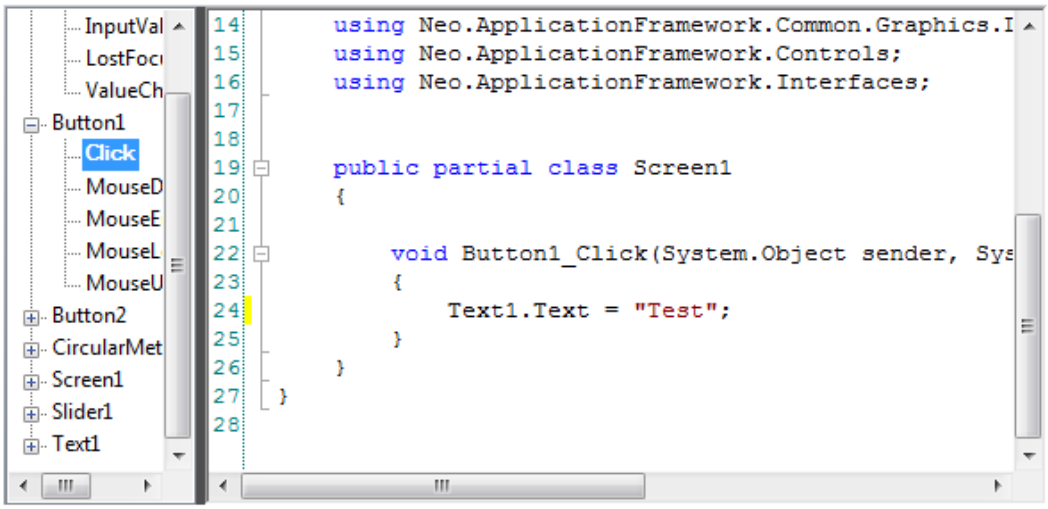
3.3.7 Käyttäjähallinta

Käyttäjiä ja salasanoja voidaan hallita iX Developerissa Security-funktion avulla. Käyttäjät voidaan jakaa eri käyttäjäryhmiin, joille voidaan määritellä erilaisia käyttöoikeuksia. Käyttöoikeuksia rajoittamalla voidaan vaikuttaa objektien näkyvyyteen tai toimintaan vaaditun käyttäjätason mukaan. Käyttöoikeudet otetaan käyttöön valitsemalla objektin Tag/Security osiosta ryhmä, jolle objektin käyttöoikeuksia halutaan rajoittaa. Tämän lisäksi voidaan vaikuttaa siihen, miten objekti käyttäytyy, jos käyttäjällä ei ole tarvittavia oikeuksia sen käyttämiseen. Objektit voidaan joko piilottaa, estää toiminnot, näyttää normaalisti tai käyttää oletuskäyttäytymistä, joka määräytyy Security -funktion asetusten mukaisesti. Sivulle, jolta kirjautuminen halutaan suorittaa, täytyy sijoittaa kirjautumistoiminto halutun objektin toimintoihin, jotta käyttäjien kirjautuminen järjestelmään olisi mahdollista.

3.3.8 Skriptit

Graafisen suunnittelun lisäksi iX Developer tukee sovelluskehitystä XAML- ja C# -ohjelmointikielillä, joiden avulla saadaan suoritettua toimintoja, joita ei muuten olisi mahdollista toteuttaa graafisella puolella. Sen lisäksi kaikki ominaisuudet ja toiminnot, jotka ovat mahdollista toteuttaa graafisesti, voidaan korvata tai suorittaa edellä mainituilla ohjelmointikielillä. IX Developerin XAML-koodituki mahdollistaa myös omien Windows Presentation Foundation -objektien luomisen ja käyttämisen omissa projekteissa.

Jokaisella ruudulla ja osilla iX Developerin funktioista on skripteille oma näkymänsä, johon päästään ruudun alavalikosta näyttötiloja vaihtamalla. Objektit, joille voidaan määritellä toimintoja, näytetään Script-näkymän vasemmassa laidassa. Objektin mahdolliset tapahtumat saadaan näkyviin laajentamalla objektin puunäkymä (KUVIO X), ja kaksoisklikkaamalla tapahtumaa iX luo kyseisen toiminnon funktiopohjan. Tämän jälkeen metodin sisään voidaan ohjelmoida normaalisti C#-ohjelmointikielellä.



Kuvio 16: Objektien toiminnot script-näkymässä

IX hyödyntää ohjelmoinnissa IntelliSense-nimentäydennystä, jonka avulla saadaan ehdotuksia kyseisen objektin ominaisuuksista ja metodeista. IntelliSense voidaan aktivoida joko manuaalisesti kirjoittamisen aikana Ctrl + Välilyönti-

näppäinyhdistelmällä tai automaattisesti koodielementtiä seuraavan pisteen jälkeen.

Skriptejä voidaan kirjoittaa iX Developerissa joko sisäisesti suoraan näyttöjen Script-näyttötilassa tai globaalisti skriptimoduuleihin, joihin vain viitataan muualta projektista. Sisäinen skripti vaikuttaa vain siinä näytössä, johon se on kirjoitettu eikä sitä voida kutsua muista näytöistä tai funktioista. Globaalisti määriteltejen mooduleiden funktiot ja muuttujat ovat käytettävissä kaikkien projektin näyttöjen välillä, ja soveltuvat parhaiten yleisesti käytettyihin funktioiden, joita kutsutaan usein projektissa. Kaikki skriptitiedostot tallentuvat .cs -tiedostoina projektihakemistoon, josta ne on helposti siirrettävissä toisiin projekteihin.

4 YHTEENVETO

Työn tavoitteena oli perehtyä iX Developeriin käyttöliittymän suunnittelussa, ja luoda tämän avulla pohjaprojekti ja modulaarisia komponentteja, jotka nopeuttaisivat uusien projektien alkuun saamista. Käyttöliittymän suunnitteluun kuului graafisen suunnittelun lisäksi elementtien toiminnallisuutta tehostavaa ohjelmointia. Alkuperäisenä tavoitteena oli saada pohjaprojektista aikaiseksi vakiolavaajan käyttöliittymä noin 10 tunnissa.

Käyttöliittymä saatiin rakennettua onnistuneesti Orfer Oy:n automaatioinsinöörien avustuksella ja yhteistyöllä, ja työ eteni alkuperäisessä aikataulussa. Suurin osa ajasta meni ohjelmiston opettelemisen jälkeen toimintaympäristön hahmottamiseen ja yhdistämiseen käyttöliittymän suunnittelun kanssa. Käyttöliittymän suunnittelu piti sisässään graafisen ja ohjelmallisen suunnittelun lisäksi projektin lataamista operaattoripaneeliin käyttäjäkokemusta ja varsinaista lavauslinjaa varten. Ohjelmointikielien osaamisen vajaus ei haitannut projektin edistämistä, sillä koodi ei ollut kovin monimutkaista. Sen sijaan ohjelmakoodin sovittaminen oikeaan ympäristöön ja teollisuusympäristön huomioon ottaminen tuotti ajoittain pieniä ongelmia. Projektin kehittämistä tapahtui kuitenkin jatkuvasti omien ideoiden kehittämisen ja parantelun lisäksi Orfer Oy:n työntekijöiden palautteen ja ehdotusten mukaisesti.

Orfer Oy tulee käyttämään työn tuloksena syntynyttä asiakasprojektia pohjana muissa vastaavanlaisissa lavausprojekteissa. Työhön liittyvä mahdollinen lisätyö tulee koostumaan lähinnä tuloratojen ja lavapaikkojen lisäämisestä projektiin, joita projektin modulaariset komponentit tukevat.

Opinnäytetyön tarjoaman mahdollisuuden avulla päästiin tutustumaan käyttöliittymäsuunnittelun vaatimuksiin ja hankaluuksiin, ja näkemään käytännössä kuinka teollisuuden lavauslinjat muodostuvat useista eri komponenteista. Työn aiheena oli käyttöliittymäsuunnittelun nopeuttaminen ja lopputuloksena syntyi suunnitellusti aihetta ja tavoitteita vastaava opinnäytetyö.

LÄHTEET

Beijer Electronics. 2012. IX Developer User's Guide [viitattu 12.04.2014]. Saatavissa: http://ftc.beijer.se/files/C125728B003AF839/87F9527143AC6CA8C1257AB4002F0BA5/iX_Developer_MAEN831H_English.pdf

Beijer Electronics. 2014. About Us [viitattu 21.03.2014]. Saatavissa: http://www.beijerelectronics.com/web/web_en_be_com.nsf/AllDocuments/E5E8BC4EB3FFF84DC1256EC10049EC78

Cogent Real-Time Systems. 2010. What is OPC [viitattu 12.04.2014]. Saatavissa: http://www.opcdatahub.com/WhatIsOPC.html

Hartson, R. & Pyla, P. S. 2012. The UX Book. USA: Elsevier.

Galitz, W. O. 2007. The Essential Guide to User Interface Design [viitattu 12.04.2014]. Indiana: Wiley Publishing. Saatavissa:

http://www.google.fi/books?id=Q3Xp_Awu49sC&lpg=PR5&ots=IZb4E-6eZ3&dq=user%20interface%20design&lr&hl=fi&pg=PA4#v=onepage&q=user%20interface%20design&f=true

IEC 62541. 2010. OPC Unified Architecture Specification [viitattu 13.04.2014]. New Jersey: Smart Grid Standards Information. Saatavissa:

https://www.smartgrid.gov/sites/default/files/doc/files/IEC_62541_OPC_Unified_Architecture_Specification_201007.pdf

Jipinghe.com. 2012. CampusEye, A Playful Interactive Installation to Explore, Discover, Share [viitattu 14.04.2014]. Saatavissa: http://jipinghe.com/2012/08/13/campuseye-capstone-project-with-panasonic-rd/

Mahnke, W. & Leitner, S-H. 2009. OPC Unified Architecture. ABB Review. 3/2009, p. 56-61 [viitattu 13.04.2014]. Saatavissa: http://www05.abb.com/global/scot/scot271.nsf/veritydisplay/75d70c47268d78bfc125762d00481f78/$file/56-61%203m903_eng72dpi.pdf

Norman, D. A. 2004. Emotional Design: Why We Love (or Hate) Everyday Things. New York: Basic Books.

Orfer Oy. 2014. Orfer Oy [viitattu 19.04.2014]. Saatavissa:

http://www.orfer.fi/suomeksi/OrferOy/tabid/5349/language/en-US/Default.aspx

Unger, R. & Chandler, C. 2012. A Project Guide to UX Design. California: New Riders